(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 March 2002 (28.03.2002)

PCT

(10) International Publication Number
WO 02/25879 A1

(51) International Patent Classification[7]: H04L 12/56

(21) International Application Number: PCT/SE01/01681

(22) International Filing Date: 26 July 2001 (26.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/666,529 20 September 2000 (20.09.2000) US

(71) Applicant: **TELEFONAKTIEBOLAGET LM ERICSSON (publ)** [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: **MIKLÓS, György**; Erzsébet u.29.fszt.1, H-1037 Budapest (HU). **FÉLEGYHÀZI, Márk**; Bolgárkertész u. 23. szt 1, H-1148 Budapest (HU).

(74) Agent: **GULLSTRAND, Malin**; Ericsson Radio Systems AB, Patent Unit Research, S-164 80 Stockholm (SE).

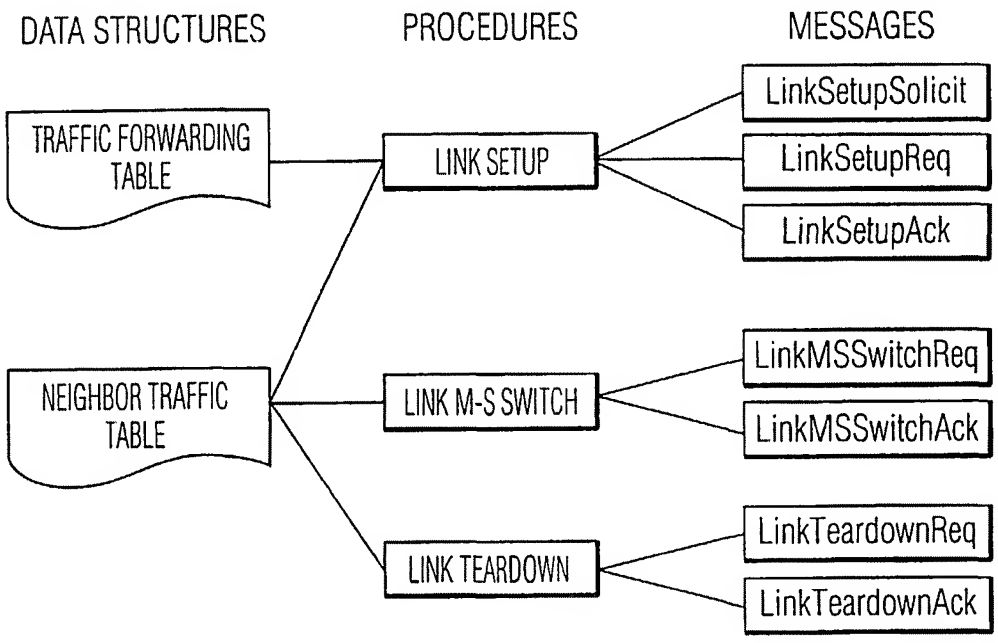
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: TRAFFIC DEPENDENT BLUETOOTH SCATTERNET OPTIMIZATION PROCEDURE



(57) Abstract: Network reconfiguration is accomplished by dynamically reconfiguring a network. Decisions whether to initiate network reconfiguration are distibuted in each node in the network, such that each node determines based upon information stored in the node whether to initiate reconfiguration of the network. Network reconfiguration can be accomplished by dynamically setting up links between nodes, dynamically tearing down links between nodes and by switching the master and slave roles between nodes.

WO 02/25879 A1

TRAFFIC DEPENDENT BLUETOOTH SCATTERNET OPTIMIZATION PROCEDURE

BACKGROUND

The present invention relates to ad-hoc networks. More particularly, the present invention relates to network optimization in ad-hoc networks.

Conventional networking protocols are based on the characteristics and/or features of networks with fixed infrastructures. In networks with fixed infrastructures, the arrangement of the links between network nodes typically does not change. The disadvantage is that networks with fixed infrastructures cannot be easily reconfigured to account for increases in data traffic, also called system loading. Accordingly, when system loading increases for one node, the surrounding nodes are likely to experience increased delays in the transmission and reception of data.

In contrast to networks with fixed infrastructures ad-hoc networks do not have a fixed infrastructures. Accordingly, nodes in ad-hoc networks act as both hosts and routers. An ad-hoc network is formed when a number of nodes decide to join together to form a network. Bluetooth is an exemplary ad-hoc networking technology. Bluetooth is an open specification for wireless communication of both voice and data. It is based on a short-range, universal radio link, and it provides a mechanism to form small ad-hoc groupings of connected devices, without a fixed network infrastructure. Bluetooth devices can include devices such as printers, PDAs, desktop computers, FAX machines, keyboards, joysticks, telephones or virtually any device. Bluetooth operates in the unlicenced 2.4 GHz Industrial-Scientific-Medical (ISM) band.

Figure 1 illustrates a Bluetooth piconet. A piconet is a collection of digital devices, such as any of those mentioned above, connected using a single Bluetooth frequency hopping channel. A piconet is initially formed with two connected

devices, herein referred to as Bluetooth nodes. A piconet can include up to eight active Bluetooth nodes. In each piconet, for example piconet 100, there exists one master Bluetooth node and one or more slave Bluetooth nodes. In figure 1 Bluetooth node 101 is the master node and node 102 is a Bluetooth slave node.

According to the Bluetooth standard a slave node can directly communicate only with the master node. However, it is assumed that Bluetooth can be implemented such that two slave nodes can communicate through the master node. Figure 2 illustrates a piconet with a master node 201 and a plurality of slave nodes 202-208 arranged in a star network topology. If slave node 202 wishes to communicate with slave node 206, slave node 202 would have to transmit the information it wished to communicate to the master node 201. Master node 201 would then transmit the information to slave node 206.

A scatternet is formed by multiple independent and unsynchronized piconets. Figure 3 illustrates an exemplary scatternet 300. In figure 3, piconet 1 includes the master node 303 and the slave nodes 301, 302 and 304; piconet 2 includes the master node 305 and the slave nodes 304, 306 and 307; piconet 3 includes the master node 309 and the slave nodes 308, 310 and 311; and piconet 4 includes master node 310 and slave node 312. As illustrated in figure 3, node 310 acts as both a slave node in piconet 3 and as the master node of piconet 4. To implement a scatternet it is necessary to use nodes which are members of more than one piconet. Such nodes are herein referred to as bridge nodes. If, for example, node 301 wishes to communicate with node 312, then nodes 304, 308 and 310 act as bridge nodes by bridging the connection between the three piconets and in particular between nodes 301 and 312. For example, node 301 transfers the information to the master node of piconet 1 node 303. Master node 303 transmits the information to bridge node 304. Bridge node 304 then forwards the information to master node 305, which in turn, transmits the information to bridge node 308. Bridge node 308 forwards the information to master node 309 which transmits the information to bridge node 310, which in turn, forwards the

information to destination node 312.

Due to the Bluetooth protocol, a Bluetooth unit can transmit or receive in only one piconet at a time. Accordingly, a bridging Bluetooth unit must switch between piconets on a time division basis. Since the bridging unit must synchronize its radio from one piconet to another and perform the necessary signaling, the throughput of the piconet is reduced due to the amount of time required to switch between piconets and due to the associated control signaling.

Although it is assumed that scatternet forming can be performed through the use of nodes participating in more than one piconet at a time as described above, the current Bluetooth specification does not define methods for selecting which links to setup and how to choose master and slave roles. As a consequence, a very large number of scatternet configurations are possible for a given set of nodes. While many different scatternets may provide connectivity for a given set of nodes, their performance in terms of achievable throughput and delay are largely different. This is because different scatternet configurations can differ in the number of hops between nodes, the amount of overhead due to bridging between piconets, and the actual traffic mix in each piconet. However, there are no known methods for using a single scatternet to optimize its own configuration.

SUMMARY

These and other problems, drawbacks and limitations of conventional techniques are overcome according to the present invention by a method and apparatus which dynamically reconfigures a network. In accordance with exemplary embodiments of the present invention the methods and apparatus are distributed in each node in the network, wherein each node determines based upon information stored in the node whether to initiate reconfiguration of the network.

In accordance with one aspect of the present invention it is determined whether a predetermined time period has expired. If the predetermined time period has expired a table is examined. A link setup procedure, a role switch procedure or a link teardown procedure based upon information in the first or

second table is selectively initiated. The link setup procedure is initiated by a node which forwards traffic between two end nodes, and the role switch procedure and the link teardown procedure are initiated by an end node.

In accordance with another aspect of the present invention a network includes a first node including a first table and a second node including a second table. A first communications link connects the first node and the second node. The first node and the second node periodically determine whether to initiate reconfiguration of the network based upon information stored in the first and second tables. The first table contains information about traffic that the first node is forwarding and information about traffic to and from nodes which are neighbors of the first node, and the second table contains information about traffic that the second node is forwarding and information about traffic to and from nodes which are neighbors of the second node.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects and advantages of the invention will be understood by reading the following detailed description in conjunction with the drawings in which:

FIG. 1 illustrates an exemplary piconet;

FIG. 2 illustrates an exemplary star-topology network;

FIG. 3 illustrates an exemplary scatternet formed by a plurality of piconets;

FIG. 4 illustrates logical elements in a node in accordance with exemplary embodiments of the present invention;

FIGs. 5A and 5B respectively illustrate an exemplary Traffic Forwarding Table and an exemplary Neighbor Traffic Table in accordance with the present invention;

FIGs. 6A and 6B illustrate a plurality of nodes which exchange messages during Link Setup in accordance with the present invention;

FIG. 7A illustrates an exemplary method for initiating Link Setup in

accordance with the present invention;

FIGs. 7B and 7C illustrates an exemplary method for performing Link Setup in accordance with the present invention;

FIGs. 8A-8C illustrate exemplary messages used for Link Setup in accordance with the present invention;

FIGs. 9A and 9B illustrate a plurality of nodes performing a Master-Slave Switch in accordance with exemplary embodiments of the present invention;

FIG. 10 illustrates an exemplary method for the initiation of the Master-Slave Switch in accordance with the present invention;

FIG. 11 illustrates an exemplary method for performing the Master-Slave Switch in accordance with the present invention;

FIGs. 12A and 12B illustrate exemplary messages for the Master-Slave Switch in accordance with the present invention;

FIGs. 13A and 13B illustrate a plurality of nodes performing a Link Teardown in accordance with exemplary embodiments of the present invention;

FIG. 14 illustrates an exemplary method for the initiation of Link Teardown in accordance with the present invention;

FIG. 15 illustrates an exemplary method for performing Link Teardown in accordance with the present invention;

FIGs. 16A and 16B illustrate exemplary messages exchanged during Link Teardown in accordance with the present invention; and

FIGS. 17-19 illustrate a plurality of nodes reconfiguring their connections using Link Setup, Master-Slave Switch and/or Link Teardown in accordance with exemplary embodiments of the present invention.

DETAILED DESCRIPTION

The present invention is directed to optimization of network configuration. In general, the present invention accomplishes this by dynamically assigning master and slave roles, and by dynamically setting up and tearing down links

between nodes. In so doing, a network topology is dynamically reconfigured based upon the current traffic requirements of the network.

In the following, the present invention is described as a technique for use in a Bluetooth scatternet. However, one skilled in the art will recognize that the present invention is applicable to other types of wireless or wireline networks.

Figure 4 illustrates the logical elements in each node in accordance with the present invention. The logical elements can be broken down into data structures, procedures and messages. Each node in the scatternet maintains a Traffic Forwarding Table and a Neighbor Traffic Table data structure. As will be described in more detail below, each node will periodically scan its Neighbor Traffic Table and Traffic Forwarding Table. Based on the information contained in these tables, the node will decide whether it may be beneficial for it to initiate a reconfiguration attempt. Link Setup procedures will be initiated by a node forwarding traffic between two other nodes, while Link Teardown and Link Master-Slave Switch procedures are initiated by one of the end nodes of a particular link.

As indicated by the links between the Traffic Forwarding Table data structure and the Link Setup procedure in figure 4, the Traffic Forwarding Table will be used by a particular node during the Link Setup procedure. Furthermore, as indicated by links between the Link Setup procedure and the LinkSetupSolicit, LinkSetupReq and LinkSetupAck messages, these messages are used during the Link Setup procedure. Moreover, as indicated by the links between the Neighbor Traffic Table data structure and Link Setup, Link Master-Slave Switch and Link Teardown procedures, the Neighbor Traffic Table will be used for these procedures. As also indicated by the links between the Link M/S switch procedure and the Link M/S switch request and LinkMSSwitchAck messages, these messages are used during the link M/S switch procedure. Finally, as indicated by the link between the LinkTeardown procedure, the LinkTeardownReq

and LinkTeardownAck messages, these messages are used during the LinkTeardown procedure.

Figure 5A illustrates an exemplary Traffic Forwarding Table in accordance with the present invention. The Traffic Forwarding Table contains information regarding data packets that are forwarded by the particular node, i.e., traffic that is neither originated from nor destined to the particular node. The first column of the Traffic Forwarding Table, "Between", indicates the pair of nodes for which the particular node is forwarding the packets. The second column of the table, "Traffic", contains a measurement of the amount of traffic that is forwarded by the particular node for each pair of nodes. The measurement of the amount of traffic contained in the second column of the Traffic Forwarding Table is an average measurement over a predetermined measurement period. The third column of the Traffic Forwarding Table, "T_{setup}", indicates the time when the last Link Setup solicit message was sent, and the fourth column, "B_{setup}", represents the backoff counter. B_{setup} is a backoff counter used for setting up a link between two nodes.

Figure 5B illustrates the Neighbor Traffic Table. The Neighbor Traffic Table contains information regarding the neighbor nodes (both current and previous neighbors) and the amount of traffic to and from each neighbor. The first column of Neighbor Traffic Table, "To Node", indicates each neighbor of the particular node, i.e., nodes that are directly connected by a Bluetooth link. In addition, nodes that are not connected to the particular node, but are visible, i.e., within radio range of the particular node, or even other nodes that are not within radio range of the particular node, can also be included in the Neighbor Traffic Table. The second column of the Neighbor Traffic Table, "Role ", indicates the role of the neighbor node, wherein an M indicates a master node role, an S indicates a slave node role, IN or OUT indicates whether the particular node is within range or out of radio range, and Unknown indicates that it is unknown whether a node is within or without of the radio range of the particular node. The

third column of the Neighbor Traffic Table, "Traffic", provides the total measured traffic on the given link in both directions. The total measured traffic is an average value over a predetermined measurement period. The fourth column of the Neighbor Traffic Table, "T_{since}", provides the time when this link was setup. The fifth column, "$T_{teardown}$", indicates the time at which the link between the particular node and the neighbor node was torn down. The sixth column, "$B_{teardown}$", is a backoff timer for the tearing down of a link between the particular node and the neighbor node. The seventh column, $T_{MSswitch}$, provides a time when the last Master-Slave Switch initiations were sent between the particular node and the neighbor node. The eighth column of the Neighbor Traffic Table, "$B_{MSswitch}$", is a backoff counter for the Master-Slave Switch.

Figures 6A and 6B illustrate a plurality of nodes which are performing a Link Setup in accordance with exemplary embodiments of the present invention. The arrows illustrated in figures 6A and 6B indicate regular traffic flow between the nodes and not necessarily the messages used for the LinkSetup procedures. Figure 6A illustrates that initially node A sends traffic which is destined for node B through node C. Node C, the forwarding node, would initiate a LinkSetup procedure and solicits nodes A and B to setup a direct link between the two nodes. Accordingly, as illustrated in Figure 6B, after the link between nodes A and B are setup, nodes A and B can directly exchange information between the two nodes without sending the information through node C. Further, as illustrated by the half-light, half-dark circle of node A, node A now is the master node of the piconet comprising nodes A and B and a slave node in the piconet comprising nodes A, B and C.

Figure 7A illustrates an exemplary method for determining whether a node should initiate a Link Setup procedure. Link reconfiguration initiation decisions are made periodically at each node at the expiration of a time period of T_{check}. This time period will not be equal at all nodes, but instead it will be chosen randomly between $T_{checkmin}$ and $T_{checkmax}$. The difference between $T_{checkmin}$ and

$T_{checkmax}$ determines the amount of randomization that will be allowed when selecting T_{check}. It will be recognized that increasing the values of $T_{checkmin}$ and $T_{checkmax}$ decreases the amount of computational and communications overhead imposed on nodes and improves the accuracy of traffic measurements and therefore makes reconfiguration attempt decisions more accurate. However, increasing these values may slow down the reconfiguration process. Accordingly, once a node, e.g., node C, determines whether the time period T_{check} has expired, the node determines whether to initiate a Link Setup procedure based upon the information stored in the Traffic Forwarding Table. Using the Traffic Forwarding Table, node C examines each row to determine whether the traffic for that row is less than $Traff_{minsetup}$. $Traff_{minsetup}$ is the minimum amount of forwarded traffic that triggers Link Setup. For each row where traffic is less than $Traff_{min\ setup}$ node C sets the backoff counter, B_{setup} equal to 1 (Step 703). Node C then selects all nodes in the Traffic Forwarding Table which have not had a Link Setup initiation for at least $P_{setup} * B_{setup}$ (Step 706). P_{setup} is a time period which is half the minimum time period between two Link Setup initiation attempts. Next node C determines whether at least one node was selected from the Traffic Forwarding Table (Step 709). If no nodes were selected from the Traffic Forwarding Table ("No" path out of decision Step 709), then the processing for node C ends (Step 712).

If at least one node from the Traffic Forwarding Table was selected ("Yes" path out of decision Step 709), then node C selects the row *R* with the highest traffic in the Traffic Forwarding Table out of the selected nodes (Step 715). Next node C determines whether the traffic in the row *R* is less than $Traff_{minsetup}$ (Step 718). If node C determines that the traffic in the selected row is less than $Traff_{minsetup}$ ("Yes" path out of decision Step 718) then the processing for the node ends (Step 712). If, however, the traffic in the selected row is not less than $Traff_{minsetup}$ ("No" path out of decision Step 718) then node C updates T_{setup} to the current time t (721). Next node C sets the backoff counter B_{setup} to a minimum

value of $2*B_{setup}$ and $B_{setupmax}$ (Step 724) and sends a LinkSetupSolicit message to any of the nodes between which traffic in the selected row *R* is forwarded (Step 727). The backoff timer B_{setup} is set to $2*B_{setup}$ to reduce the frequency that a node initiates a Link Setup, while setting an upper limit of $B_{setupmax}$ prevents the backoff timer from increasing to a very high value where Link Setup is initiated too infrequently.

Figure 7B illustrates an exemplary method for a node which has received a LinkSetupSolicit message. Initially, a node, e.g., node A, receives a LinkSetupSolicit message from C (Step 730). Figure 8A illustrates the LinkSetupSolicit message which includes a From field, a To field, a Between field and a WinNet field. The From field indicates the node which is sending the LinkSetupSolicit message, the To field indicates the node which is to receive the LinkSetupSolicit message, the Between field indicates the two end points of the new link and the WinNet field indicates the amount of capacity in kilobits per second (kbps) that is expected to be freed at the forwarding node as a result of rerouting the forwarded traffic to the new link.

Node A then determines whether a time period of at least P_{down} has passed since the last link was torn down (Step 733). The time period P_{down} is a constant value and is selected such that nodes which have recently been setup are not torn down. It will be recognized that increasing the value of P_{down} slows the reconfiguration process but decreases the risk of unnecessarily setting up a new link that will be torn down soon afterwards. If the node determines that P_{down} has not passed ("No" path out of decision Step 733) then the Link Setup procedure fails (Step 736). If, however, the time period P_{down} has passed ("Yes" path out of decision Step 710) then node A determines the amount of capacity increase WinMaster(A) and WinSlave(A) if the requested link were built (Step 739). It will be recognized that the increase in capacity is equal to the negative of the increase in the overhead. If both WinNet + WinSlave(A) and WinNet + WinMaster(A) are less than zero ("Yes" path out of decision Step 742) then the

procedure fails (Step 736). If WinNet + WinSlave(A) and WinNet + WinMaster(A) are not both less than zero ("No" path out of decision Step 742) then node A sends a LinkSetupReq message to the other end of the link including WinMaster(A) and WinSlave(A) (Step 745).

Figure 8B illustrates the LinkSetupReq message which includes a From field, a To field, a Traff field, an M/S field, a WinNet field, a WinMaster field and a WinSlave field. The From field indicates the node which is sending the LinkSetupReq message, the To field indicates the link which is to receive the LinkSetupReq message and the Traff field is the amount of current traffic activity for the source node including both incoming and outgoing traffic. The M/S field indicates whether the node which is sending the LinkSetupReq message is a master or not, the WinNet field is the amount of capacity that the forwarding node is expected to gain as a result of setting up the link, the WinMaster field and the WinSlave field are the amount of capacity that the source is expected to gain as a result of setting up the link when the source node acts as the master or a slave node, respectively. Accordingly, in the example described above in connection with figure 7B, node A would place WinMaster(A) in the WinMaster field and WinSlave(A) in the WinSlave field.

Figure 7C illustrates an exemplary method for a node which receives a LinkSetupReq message. A node, e.g., node B, receives the LinkSetupReq message (Step 748) and calculates WinMaster(B) and WinSlave(B) (Step 751). Next node B calculates gain M, which is equal to WinNet + WinMaster(B) + WinSlave(A) and gain S, which is equal to WinNet + WinMaster(A) + WinSlave(B) (Step 754). If gain M and gain S are both greater than 0 ("Yes" path out of decision step 757) then node B selects the larger of the gains between gain M and gain S and sends a LinkSetupAck message to node A (Step 760). If gain M is larger than Gain S then node B acts as the master node and if Gain S is larger than Gain M node B acts as a slave node.

Figure 8C illustrates the LinkSetupAck message which includes a From

field, a To field, an M/S field and an Info field. The From field indicates the node which is sending the Link Setup acknowledge message, the To field indicates the destination of the Link Setup acknowledge message, the M/S field indicates whether the node which is sending the Link Setup acknowledge message will become the master or slave of the new link and the Info field can contain additional information which may be used in the actual Link Setup process, e.g., page timing information.

If gain M and gain S are not both greater than 0 ("No" path out of decision step 757) then node B determines whether gain M is greater than 0 (Step 763). If gain M is greater than 0 ("Yes" path out of decision step 763) then node B sends accepts the setup request with node B acting as the master node by sending a LinkSetupAck message (Step 766). If the gain M is not greater than zero ("No" path out of decision step 763) node B determines whether the gain S is greater than zero (Step 769). If the gain S is greater than zero ("Yes" path out of decision Step 769) then node B accepts the setup request with node B acting as a slave node by sending a LinkSetupAck message (Step 772). If, however, the gain S is not greater than zero ("No" path out of decision Step 769) then node B will refuse the request (Step 775). Node B refuses the request by not responding to node A with an acknowledgment message. Accordingly, the load on the network will not be increased when requests to setup links are refused.

The procedure described above in connection with figure 7A-7C begins with the solicitation of a new link between the two nodes that create the highest amount of forwarded traffic. If this procedure is successful, the traffic measurement will drop below the threshold $Traff_{min}$ setup before the new check $2*P_{setup}$ time later. In addition, an exponential backoff scheme is used for sending solicitation messages in order to avoid a high load of these control messages. The exponential backoff scheme is implemented such that the backoff counter is multiplied by two each Link Setup attempt, and hence, unsuccessful Link Setup attempts are repeated with half the frequency of the previous Link Setup attempt.

Further, although the procedure described above in connection with figure 7 illustrates selecting the larger of the Gain M and Gain S, the procedure can also be implemented where if one of the nodes is already the master and the other node is not then the master node will also be the master of the new link. If neither of the nodes is the master or both of them are, then the node with the highest traffic activity will be the master of the new link.

Figures 9A and 9B illustrate a plurality of nodes before and after a Master-Slave Switch. The arrows illustrated in figures 9A and 9B indicate regular traffic flow between the nodes and not necessarily the messages used for the Master-Slave Switch procedures. Accordingly, as illustrated in Figure 9A, node A is the master of the piconet comprising nodes A and B, and node A is a slave in the piconet comprising nodes A, B and C. Since node A is exchanging information with node B and since node A is both the master of one piconet and a slave of another piconet, node A must divide its time between the piconet comprising nodes A and B and the piconet comprising nodes A, B and C. Accordingly, node A determines that it is more beneficial for it to play the master role in the link between nodes A and B. As illustrated in Figure 9B, after the Master-Slave Switch is performed between nodes A and C, node A has now become the master of the piconet comprising nodes A, B and C. Further, node C is a slave in the piconet comprising nodes A and C and the master of the piconet comprising nodes B and C. By playing the master role in both links node A can increase its throughput of information being transmitted to node B because node A does not have to switch to participate in the piconet comprising nodes A, B and C.

Figure 10 illustrates an exemplary method for initiating a Master-Slave Switch. As discussed above, Master-Slave Switches are only initiated by end nodes, i.e., either a source or destination node, and are based upon information contained in one of the end node's Neighbor Traffic Table. If, an end node, e.g., node A, determines that a time period T_{check} has expired the node then determines whether it is a member of only one piconet (Step 1010). If node A is a member of

only one piconet ("Yes" path out of decision Step 1010) then the Master-Slave Switch procedure for node A ends (Step 1020). If, however, node A is a member of more than one piconet ("No" path out of decision Step 1010) then node A selects all nodes in the Neighbor Traffic Table which have not had a Link Setup initiation attempt for at least the time period of $P_{MSSwitch} * B_{MSSwitch}$ (Step 1025). $P_{MSSwitch}$ is half the minimum time period between Master-Slave Switch initiation attempts.

Next, node A determines whether at least one node was selected (Step 1030). If no nodes were selected ("No" path out of decision step 1030) then processing for node A ends (Step 1020). If, however, at least one node was selected ("Yes" path out of decision step 1030) then node A selects the row *R* with the highest traffic in the neighbor traffic table out of the selected nodes (Step 1035). Next, node A determines whether the traffic in the row *R* is equal to zero (Step 1040). If the traffic in the row *R* is equal to zero ("Yes" path out of decision Step 1040) then the Master-Slave Switch procedure ends for node A (Step 1020).

If the traffic in the row *R* is not equal to zero ("No" path out of decision Step 1040) then node A updates $T_{MSSwitch}$ to the current time t (Step 1050) and sets the backoff timer $B_{MSSwitch}$ equal to a minimum value of $2*B_{MSSwitch}$ and $B_{MSSwitchmax}$ (Step 1060). The backoff timer $B_{MSSwitch}$ is set to $2*B_{MSSwitch}$ to reduce the frequency that a node initiates a Master-Slave Switch, while an upper limit of $B_{MSSwitchmax}$ prevents the backoff timer from increasing to a very high value where a Master-Slave Switch is initiated too infrequently. Next node A sends a LinkMSSwitchReq message to node B (Step 1070). Figure 12A illustrates a LinkMSSwitchReq message. The link MS switch request message includes a From field, a To field, a Traff field, an M/S field and a Win field. The From field indicates the source of the LinkMSSwitchReq message, the To field indicates the destination node of the LinkMSSwitchReq message, the Traff field indicates the amount of traffic activity on the source node, the M/S field indicates whether

the source node is a master node and the Win field is the amount of capacity that the source node expects to gain as a result of a Master-Slave Switch.

Figure 11 illustrates an exemplary method for a node which receives a LinkMSSwitchReq message. Initially, node B receives the LinkMSSwitchReq message (Step 1130) and calculates Win(B) (Step 1140). Node B then determines whether Win(A) + Win(B) is greater than or equal to zero (Step 1150). If node B determines that Win(A) + Win(B) is not greater than or equal to zero ("No" path out of decision Step 1150) then node B will refuse the switch (Step 1160). To refuse the switch node B does not do anything, i.e., node B does not send a message to node A indicating that node B is refusing the switch. By not sending a message, the load on the network will not be increased by sending messages for refusing Link Master-Slave Switch request. If node B determines that Win(A) + Win(B) is greater than or equal to zero ("Yes" path out of decision Step 1150) then node B determines whether Win(A) + Win(B) is equal to zero (Step 1170). If it is determined that Win(A) + Win(B) is not equal to zero ("No" path out of decision Step 1170) then node B performs the Master-Slave Switch and sends a LinkMSSwitchAck message to node A (Step 1180).

Figure 12B illustrates a LinkMSSwitchAck message. The LinkMSSwitchAck message includes a From field, a To field and an Info field. The From field indicates the source node of the LinkMSSwitchAck message, the To field indicates the destination node of the LinkMSSwitchAck message and the Info field can contain optional additional information, e.g., timing information, used for performing the Master-Slave Switch.

If it is determined that Win(A) + Win(B) is equal to zero ("Yes" path out of decision Step 1170) then node B determines whether node A has a higher traffic activity than node B (Step 1190). If node B determines that node A has a higher traffic activity than node B ("Yes" path out of decision Step 1190) then node B and node A will perform the Master-Slave Switch and node B will send a LinkMSSwitchAck message to node A (Step 1180). If, however, node B

determines that node A does not have a higher traffic activity ("No" path out of decision Step 1190) then node B refuses the Master-Slave Switch (Step 1160).

Figures 13A and 13B illustrate a plurality of nodes before and after the Link Teardown procedure has been performed. The arrows illustrated in figures 13A and 13B indicate regular traffic flow of messages between the nodes and not necessarily the messages used for the Link Teardown procedures. As illustrated in Figure 13A, node A is the master of the piconet comprising nodes A, B and C, and node C is the master of a piconet comprising nodes C and B. As illustrated in Figure 13B, it is determined that the link between nodes B and C should be torn down. Accordingly, the link between nodes B and C is torn down as illustrated in Figure 13B.

Figure 14 illustrates an exemplary method for determining whether to initiate a Link Teardown procedure in accordance with the present invention. Link Teardown procedures are initiated by an end node based upon information in the Neighbor Traffic Table. After determining that a time period of T_{check} has expired, the node performing the Link Teardown initiation procedure, e.g., node A, computes an estimated capacity increase (Win) and network capacity decrease (WinNet) for each link if torn down (Step 1410), wherein WinNet=-2*Traf and Traf is the traffic on the Link. Next node A selects the rows in the Neighbor Traffic Table which have links which are up for at least P_{up} and which have not had a Link Teardown initiation attempt for at least $P_{setup} * B_{teardown}$ (Step 1415). P_{up} is the minimum time period before a link may be torn down after the link has been setup. Next node A determines whether at least one row was selected (Step 1420). If no row was selected ("No" path out of decision Step 1420) the processing for node A ends (step 1425). If at least one row was selected ("Yes" path out of decision Step 1420) then node A selects the row *R* with the highest value of Win+WinNet out of the selected rows (Step 1430) and determines whether the value of Win+WinNet is less than zero (step 1440). If it is determined that the value of Win + WinNet is less than zero ("Yes" path out of

decision Step 1440) then the processing for node A ends (Step 1425).

If node A determines that the value of Win + WinNet is not less than zero ("No" path out of decision Step 1440) then the node updates $T_{teardown}$ to the current time t (Step 1450) and sets the backoff timer $B_{teardown}$ to a minimum of $2*B_{teardown}$ and $B_{teardownmax}$ (Step 1460). The backoff timer $B_{teardown}$ is set to $2*B_{teardown}$ to reduce the frequency that a node initiates a Link Teardown, while an upper limit of $B_{teardownmax}$ prevents the backoff timer from increasing to a very high value where a Link Teardown is initiated too infrequently. Next node A calculates Win(A) (Step 1465) and sends a LinkTeardownReq message to the node with which it wishes to teardown the link (Step 1470).

Figure 16A illustrates an exemplary LinkTeardownReq message in accordance with the present invention. The LinkTeardownReq message includes a From field, a To field, a Traff field, a Win field and a WinNet field. The From field indicates the node sending the LinkTeardownReq message, the To field indicates the destination of the LinkTeardownReq message and the Traff field indicates the traffic activity at the source node. The Win field contains an indication of the amount of capacity that the source node is expected to gain as a result of the Link Teardown and the WinNet field is the estimated negative gain in network capacity as a result of the Link Teardown, i.e., WinNet in this case represents the increase in overhead due to the Link Teardown.

Figure 15 illustrates an exemplary Link Teardown procedure in accordance with the present invention. Initially, node B receives the LinkTeardownReq message (Step 1530) and calculates Win (B) and WinNet (Step 1540). Node B then determines whether Win(A) + Win(B) + WinNet is greater than or equal to zero (Step 1550). If node B determines that Win (A) + Win (B) + WinNet is not greater than or equal to zero ("No" path out of decision Step 1550) then node B refuses the LinkTeardownReq (Step 1560). In accordance with exemplary embodiments of the present invention, node B refuses the LinkTeardownReq by not doing anything, i.e., the node B does not send a message to node A indicating

that node B is refusing the request. By not sending a message, the load on the network will not be increased by sending messages for refusing a Link Teardown request. If, however, node B determines that Win (A) + Win (B) + WinNet is greater than or equal to zero ("Yes" path out of decision Step 1550) then node B accepts the LinkTeardownReq and sends a LinkTeardownAck message to node A (Step 1570).

Figure 16B illustrates an exemplary LinkTeardownAck message in accordance with the present invention. The LinkTeardownAck message includes a From field indicating the source of the LinkTeardownAck, a To field indicating the destination of the LinkTeardownAck message and an Info field containing optional information, e.g., timing information.

In accordance with the present invention, the sending of the LinkTeardownAck message triggers route maintenance procedures of the routing protocol such that the direct link between the source and destination nodes, i.e., the link To and From nodes, will be marked as unavailable. If the routing protocol finds an alternative path the LinkTeardownAck message can be sent on this alternative path. Otherwise, if the routing protocol fails to find an alternative path an error will result. In this case the LinkTeardownAck message is dropped and the link is logically restored as available to the network. If, however, an alternative path is found, when the LinkTeardownAck message arrives on the alternative path to the other end the link may be torndown.

To avoid the link being logically marked as unavailable and then restored because of a dropped LinkTeardownAck message, the originator of the LinkTeardownReq message may trigger an alternative path discovery procedure before sending the message. Such a procedure may be part of the routing protocol, or it can be implemented as an addition to the routing protocol. As a result of the alternative path discovery, the node will not send the request when such an alternative path is not available. When such an alternative path is available, it can be possible to determine the length of the alternative path,

HopCount. By sending the LinkTeardownReq message only when an alternative path has been found avoids the other end node unnecessarily marking the link as unavailable causing temporary disruptions on the traffic flowing on the link. Further, knowledge of the alternative path can be used to estimate the load on the network that will be caused by tearing the link down, i.e., WinNet. Using the HopCount, WinNet can be estimated using the formula WinNet=-2*(HopCount-1)*Tr, wherein Tr is the traffic on the link.

If two or more Link Teardown attempts are being performed simultaneously, one Link Teardown procedure may teardown a link being used by other Link Teardown procedures. To avoid two or more Link Teardown attempts being performed simultaneously and periodically that would block alternative paths to each other, Link Teardown initiation should be performed at randomized time instants.

In accordance with exemplary embodiments of the present invention, since the Link Setup procedure uses a different table than the Link Teardown and the Master-Slave Switch procedures, the Link Setup procedure can be performed in parallel with the Link Teardown and the Master-Slave Switch procedures. Alternatively, Link Setup procedures can be performed serially with the Link Teardown and the Master-Slave Switch procedures. In addition, the present invention may be implemented such that a node initially determines whether a link can be torndown and if the Link Teardown procedures are unsuccessful then the node would initiate the Master-Slave Switch procedures.

Now that the exemplary procedures of the present invention have been described, several examples will be presented to further illustrate the advantages of the present invention. Figures 17A through 17F illustrate a plurality of nodes where a node which is an application server is not the master of the piconet. The arrows illustrated in figures 17A through 17F indicate regular traffic flow between the nodes and not necessarily the messages used for the network reconfiguration procedures. Accordingly, Figure 17A illustrates nodes A, B, C and D, where

node C is the master of the piconet while node A is an application server. Since node A is an application server, the nodes of the piconet, i.e., nodes B through D, will be requesting information from node A. However, since node C is the master of the piconet, all traffic from node A to nodes B and D must pass through node C. This results in an inefficient throughput of the piconet. By measuring the amount of traffic it is forwarding node C recognizes that the throughput of the piconet is inefficient and sends a LinkSetupSolicit message to node A. Node A then sends a LinkSetupReq message through node C to node B and node B acknowledges the setup of a link between nodes A and B by sending a LinkSetupAck message to node A through node C. Accordingly, a new link is setup between nodes A and B as illustrated in Figure 17B.

Figure 17B illustrates the teardown of a link between nodes B and C. Initially, node B will send a LinkTeardownReq to node C requesting the teardown of the B-C link. In response, node C sends a LinkTeardownAck message acknowledging the teardown of the link between nodes B and C. Hence, the link between nodes B and C is torn down.

Figure 17C illustrates a Link Setup between nodes A and D. The Link Setup is initiated by node C sending a LinkSetupSolicit message to node A. Node A responds with a LinkSetupReq message which is sent through node C to node D. Node D accepting the Link Setup sends a LinkSetupAck message through node C to node A acknowledging the setup of a link between nodes A and D.

Figure 17D illustrates the teardown of the link between nodes C and D. Initially, node D sends a LinkTeardownReq message to node C requesting to teardown the C-D link. Node C accepts the Link Teardown and sends a LinkTeardownAck message to node D acknowledging the teardown of the C-D link.

Figure 17E illustrates the LinkMSSwitch between nodes A and C. As illustrated in Figure 17E, node A is a slave of the piconet comprising nodes A and C, and node A is the master of the piconet comprising nodes A, B and D.

Accordingly, node A sends node C a LinkMSSwitchReq message requesting to switch the master and slave roles between the two nodes. Node C acknowledges the Master-Slave Switch by sending a LinkMSSwitchAck message back to node A.

Figure 17F illustrates the network after it has been optimized by the above-described procedure. Node A which is an application server is now the master of the piconet comprising nodes A, B, C and D. Accordingly, the throughput of this piconet has now been increased by allowing an application server to be the master of a piconet in which the nodes which are using the applications are slaves.

Figures 18A through 18D illustrate another example of a plurality of nodes which switch from an inefficient network into a more efficient network using exemplary procedures of the present invention. The arrows illustrated in figures 18A through 18D indicate regular traffic flow between the nodes and not necessarily the messages used for the network reconfiguration procedures. In Figures 18A through 18D node A is an application server with nodes B through D as clients of the application server. As illustrated in Figure 18A, node A, an application server, is a slave in a piconet comprising nodes A and B; a slave in a piconet comprising nodes A and C; and a slave in a piconet comprising nodes A and D. Since an application server, node A, is a slave in three piconets, the network is very inefficient due to the overhead of node A switching from one piconet to another. Recognizing this situation, node A sends a LinkMSSwitchReq message to node C requesting that node A become the master of the piconet comprising nodes A and C. Node C accepts the Master-Slave Switch and sends a LinkMSSwitchAck message back to node A.

Figure 18B illustrates that node A is now the master of the piconet comprising nodes A and C and a slave of the piconet comprising nodes A and B and the piconet comprises nodes A and D. Accordingly, node A sends a LinkMSSwitchReq message to node D requesting that node A become the master of the piconet comprising nodes A and D. Accepting the Master-Slave Switch, node D sends a LinkMSSwitchAck message back to node A.

Figure 18C illustrates that node A is now the master of the piconets comprising nodes A, C and D, and the master of the piconet comprising nodes A and D, and a slave in the piconet comprising nodes A and B. Node A initiates a Master-Slave Switch by sending a LinkMSSwitchReq message to node B. Node B accepts the Master-Slave Switch and sends a LinkMSSwitchAck message back to node A. Accordingly, as illustrated in Figure 18D, an application server, node A, is the master of the piconet including nodes B through D, which results in a more efficient network configuration.

Figures 19A through 19I illustrate a plurality of nodes which are initially connected as a string and eventually connected in a star configuration. The arrows illustrated in figures 19A through 19I indicate regular traffic flow between the nodes and not necessarily the messages used for the network reconfiguration procedures. Further, node A is an application server sending information to nodes B through F which act as clients. In Figure 19A, node B is the master of a piconet comprising nodes A, B and C; node D is the master of the piconet comprising nodes C, D and E; and node F is the master of a piconet comprising nodes F and E. Based upon the configuration illustrated in Figure 19A, node B requests that a link be setup between nodes A and C, node C requests that a link be setup between nodes B and D, and node E requests that a link be setup between nodes D and F.

Figure 19B illustrates the results of the various Link Setups described in connection with Figure 19A. Accordingly, node B is the master of a piconet comprising nodes A, B, C and D; node C is the master of a piconet comprising nodes C and A; node D is the master of a piconet comprising nodes C, D, E and F; and node F is the master of a piconet comprising nodes E and F. Based upon the configuration illustrated in Figure 19B, C performs a Link Teardown with node B, D performs a Link Teardown with node B, and node E performs a Link Teardown with node F.

Figure 19C illustrates the results of the various Link Teardowns described

above in connection with Figure 19B. As illustrated in Figure 19C, node B is the master of a piconet comprising nodes A and B; node C is the master of a piconet comprising nodes A and C; and node D is the master of a piconet comprising nodes D, C, E and F. Based upon the configuration illustrated in Figure 19C, node C performs a Master-Slave Switch with node D and node A performs a Master-Slave Switch with node B.

Figure 19D illustrates the results of the Master-Slave Switches described above in connection with Figure 19C. As illustrated in Figure 19D, node A is the master of a piconet comprising nodes A and B; node C is the master of a piconet comprising nodes A, C and D; and node D is the master of a piconet comprising nodes D, E and F. Based upon the configuration illustrated in Figure 19D, node C initiates a Link Setup procedure between nodes A and D and node D initiates a Link Setup procedure between nodes C and F.

Figure 19E illustrates the result of the above-described Link Setup procedures. As illustrated in Figure 19E, node A is the master of a piconet comprising nodes A, B and D; node C is the master of a piconet comprising nodes A, C, D and F; and node D is the master of a piconet comprising nodes D, E and F. Based upon the configuration illustrated in Figure 19E, node D performs a Link Teardown procedure with node C and node F performs a Link Teardown procedure with node D.

Figure 19F illustrates the results of the Link Teardown procedures described above in connection with Figure 19E. In Figure 19F, node A is the master of a piconet comprising nodes A, B and D; node C is the master of a piconet comprising nodes A, C and F; and node D is the master of a piconet with node E. Based upon the configuration illustrated in Figure 19F, node A initiates a Master-Slave Switch with node C.

Figure 19G illustrates the results of the Master-Slave Switch between nodes A and C described above in connection with Figure 19F. As illustrated in Figure 19G, node A is the master of a piconet comprising nodes A, B, C and D;

node C is the master of a piconet comprising nodes C and F; and node D is the master of a piconet comprising nodes D and E. Based upon the configuration illustrated in Figure 19G, node C initiates a Link Setup procedure between nodes A and F and node D initiates a Link Setup procedure between nodes A and E.

Figure 19H illustrates the results of the Link Setup procedures described above in connection with Figure 19G. As illustrated in Figure 19H, node A is the master of a piconet including nodes A, B, C, D, E and F; node C is the master of a piconet comprising nodes C and F; and node D is the master of a piconet comprising nodes D and E. Based upon the configuration illustrated in Figure 19H, node F initiates a Link Teardown procedure with node C and node D initiates a Link Teardown procedure with node E.

Figure 19I illustrates the result of the Link Teardown procedures described above in Figure 19H. As illustrated in Figure 19I, node A, which is an application server, is now the master of a piconet comprising nodes A through F. Accordingly, nodes B through F can now directly request and access information stored on the application server A without having to send the request or receive the information through other nodes.

It will be recognized that in certain situations using the above-described network reconfiguration procedures may not result in the most efficient configuration of the network being formed because an intermediate configuration results in worse performance than the original configuration. For example, referring now to figures 20A-20C, assume that the network configuration of 20C is more efficient than the network configurations of figures 20A and 20B. Further assume that the network configuration of figure 20A is more efficient than the network configuration of figure 20B. Since the network configuration illustrated in figure 20A is more efficient than the network configuration in figure 20B, the nodes will not perform a reconfiguration to the network configuration in figure 20B, and hence, the more efficient network configuration of figure 20C will not be formed. In other words, since figure 20B has more links, the nodes may

interpret this as resulting in higher overhead. To address this situation, the requirements for Link Setup conditions can be loosened. For example, more optimistic, i.e., smaller, values for the overhead of establishing a new link when computing the Win values in the Link Setup procedure can be used. If the smaller values of the overhead for establishing a new link are used a longer interval for P_{down} should be used to avoid instability which would result from setting up a new link that has been recently torn down.

Although the present invention has been described using a constant time period for a node to initiate reconfiguration attempts, a node could set the time between sending two reconfiguration attempt messages based on the amount of traffic that would be affected by the reconfiguration. Accordingly, a reconfiguration which affects a lot of traffic will likely be performed earlier than another reconfiguration which affects less traffic. Adjusting the time between reconfiguration attempts can be useful when a number of nodes perform reconfiguration attempts simultaneously and these reconfiguration attempts exclude each other. For example, if two nodes independently initiate a new Link Setup, where one of the node is forwarding 50kbps of information and the other node is forwarding 200kbps of information. When the two Link Setups exclude each other, i.e., one of the Link Setup will not be performed as soon as the other Link Setup is performed, then it is beneficial if the one forwarding the less amount of traffic waits a longer time before initiation than the other. In this way it is more likely that a reconfiguration that affects more traffic will be the one to be carried out.

Although in the procedures above, a reconfiguration attempt would not be performed if the expected capacity change is negative, i.e., there will be less capacity after the reconfiguration than before, in certain situations it may be advantageous to allow these reconfiguration to occur. For example, by taking into account the amount of free capacity that a node has, a node which has a large amount of free resources can accept a reconfiguration attempt which implies a

higher amount of overhead because this does not reduce the network capacity if more overhead is imposed on nodes that have free capacity. Further, network capacity can be improved if a reconfiguration decreases the overhead at overloaded nodes even at the cost of imposing more overhead on nodes with free capacity.

Further, although the present invention has been described as using two separate tables, i.e., a Traffic Forwarding Table and a Neighbor Traffic Table, to make reconfiguration decisions, it will be recognized that these tables can be combined into one table.

The present invention has been described with reference to several exemplary embodiments. However, it will be readily apparent to those skilled in the art that it is possible to embody the invention in specific forms other than those of the exemplary embodiments described above. This may be done without departing from the spirit of the invention. These exemplary embodiments are merely illustrative and should not be considered restrictive in any way. The scope of the invention is given by the appended claims, rather than the preceding description, and all variations and equivalents which fall within the range of the claims are intended to be embraced therein.

WHAT IS CLAIMED IS:

1. In an ad-hoc network a method for reconfiguring the network comprising the steps of:

determining whether a predetermined time period has expired;

examining a table if the predetermined time period has expired; and

selectively initiating a link setup procedure, a role switch procedure or a link teardown procedure based upon information in the first or second table,

wherein the link setup procedure is initiated by a node which forwards traffic between two end nodes, and the role switch procedure and the link teardown procedure are initiated by an end node.

2. The method of claim 1, wherein the first table contains information about traffic that a node is forwarding and the second table contains information about traffic of neighboring nodes.

3. The method of claim 1, wherein the link setup procedure comprises the steps of:

sending a link setup solicit message to a first end node;

sending a link setup request message from the first end node to a second end node; and

responding with a link setup acknowledgment message from the second end node if the second end node accepts the link setup.

4. The method of claim 3, wherein the second node accepts the link setup based upon whether the link to be setup increases the capacity of the network.

5. The method of claim 1, wherein the link role switch procedure

comprises the steps of:

sending a link role switch request message from a first end node to a second end node; and

responding with a link role switch acknowledgment message from the second node to the first node if the second node accepts the role switch.

6. The method of claim 5, wherein the second node accepts the role switch if the role switch increases the network capacity or if the network capacity is unchanged by the role switch, the second node accepts the role switch if the first node has a higher traffic activity than the second node.

7. The method of claim 1, wherein the link teardown procedure comprises the steps of:

sending a link teardown request message from a first node to a second node; and

responding with a link teardown acknowledgment message from the second node to the first node if the second node accepts the link teardown request.

8. The method of claim 7, wherein the second node accepts the link teardown request based upon the sum of capacity increase at the first and second nodes as well as a capacity change in the network due to rerouting of traffic from the torn down link.

9. The method of claim 1, wherein the network operates in accordance with Bluetooth protocol.

10. The method of claim 1, wherein the network is a wireless network.

11. The method of claim 7, wherein the link teardown procedure

further comprises the steps of:

triggering a route maintenance procedure to find an alternative path between the first node and the second node; and

sending the link teardown acknowledgment message over the alternative path.

12. The method of claim 7, wherein the link teardown procedure further comprises the steps of:

triggering a route maintenance procedure at the first node to find an alternative path prior to sending the link teardown request message; and

sending the link teardown request message to the second node if the an alternative path is found.

13. The method of claim 1, wherein the predetermined time period is set based upon an amount of traffic that would be affected by the reconfiguration.

14. The method of claim 1, wherein if the link setup procedure, role switch procedure or the link teardown procedure fails, performing the step of:

setting a backoff timer to a minimum value between two times a current value of the backoff timer and a maximum backoff timer value.

15. A network comprising:

a first node including a first table;

a second node including a second table; and

a first communications link connecting the first node and the second node,

wherein the first node periodically determines whether to initiate reconfiguration of the network based upon information stored in the first table and the second node periodically determines whether to initiate reconfiguration of the network based upon information stored in the second table.

16. The network of claim 15, further comprising:

a third node comprising a third table; and

a second communications link connecting the third node to the second node,

wherein the third node periodically determines whether to initiate reconfiguration of the network based upon information stored in the third table.

17. The network of claim 16, wherein the second node periodically determines, based upon an amount of traffic forwarded by the second node between the first and third nodes, whether to initiate a link setup procedure between the first and third nodes.

18. The network of claim 17, wherein the link configuration procedure establishes a third communications link between the first node and the third node.

19. The network of claim 15, wherein the first node, based upon an amount of traffic handled by the first node and an amount of capacity that the first node estimates it will be able to gain by a master-slave switch, initiates a master-slave switch procedure with the second node.

20. The network of claim 15, wherein the first node, based upon an amount of capacity that the first node estimates it will be able to gain as the result of a link teardown and based upon an amount of capacity that the network will gain as a result of the link teardown, initiates a link teardown procedure with the second node.

21. The network of claim 15, wherein first table contains information about traffic that the first node is forwarding and information about traffic to and from nodes which are neighbors of the first node, and the second table contains

information about traffic that the second node is forwarding and information about traffic to and from nodes which are neighbors of the second node.

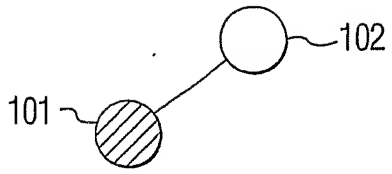


FIG. 1

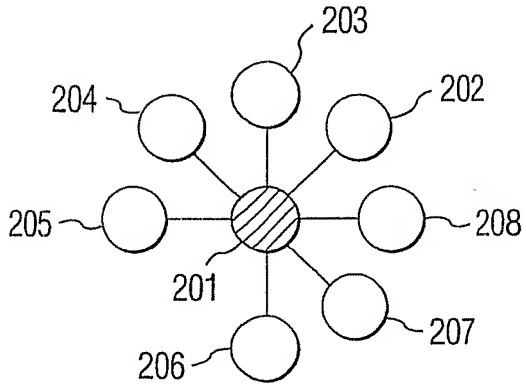


FIG. 2

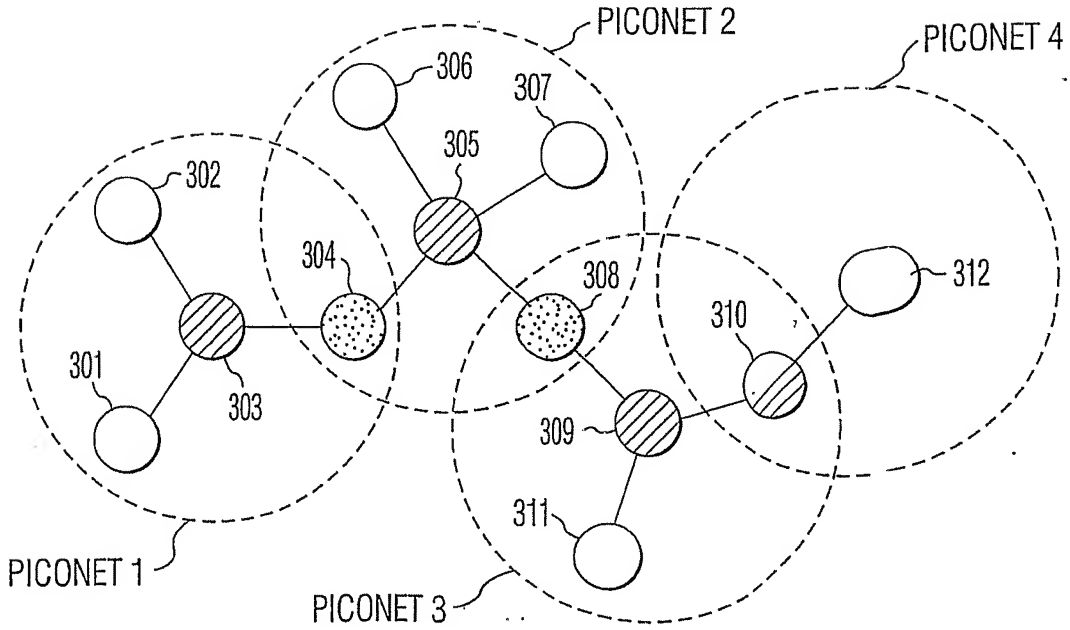


FIG. 3

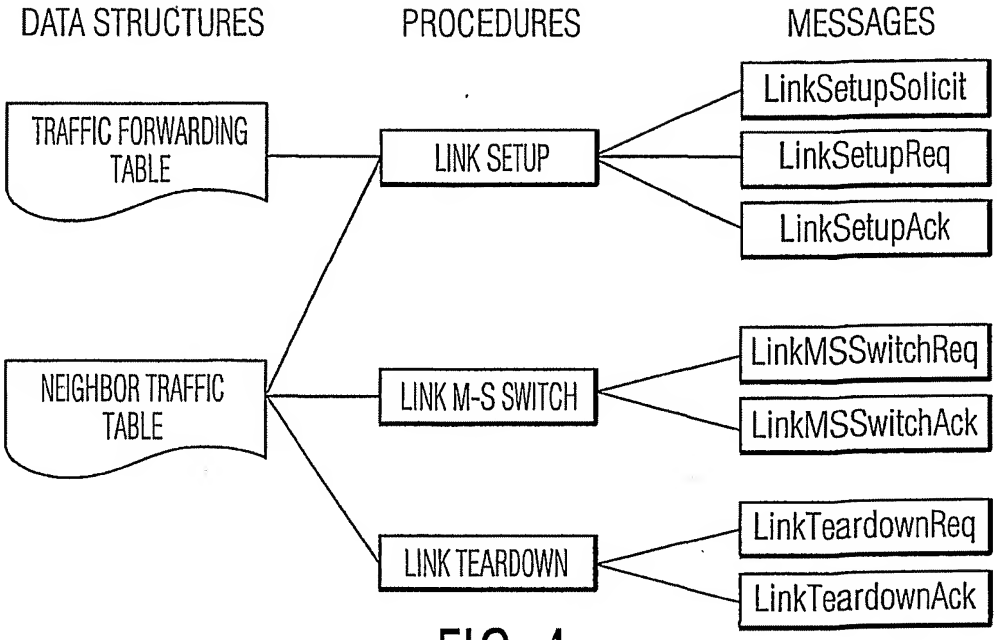


FIG. 4

BETWEEN	TRAFFIC	T_{setup}	B_{setup}
B-C	100	60	1
B-D	20		1
C-B	5		1
C-E	0		1
C-F	0		1
C-G	0		1

FIG. 5A

TO NODE	ROLE	TRAFFIC	T_{Since}	$T_{teardown}$	$B_{teardown}$	$T_{MSswitch}$	$B_{MSswitch}$
B	M	100	120		1		1
C	M	10	110		1		1
D	S	15	80	150	2		1
E	-/IN	-	-				
F	-/OUT	-	40				
G	-/unknown	-	90				

FIG. 5B

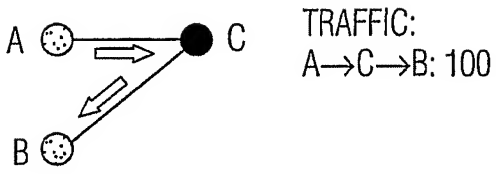


FIG. 6A

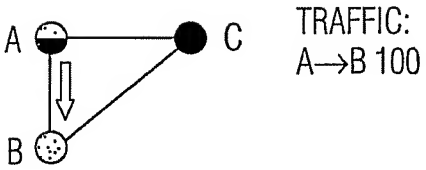


FIG. 6B

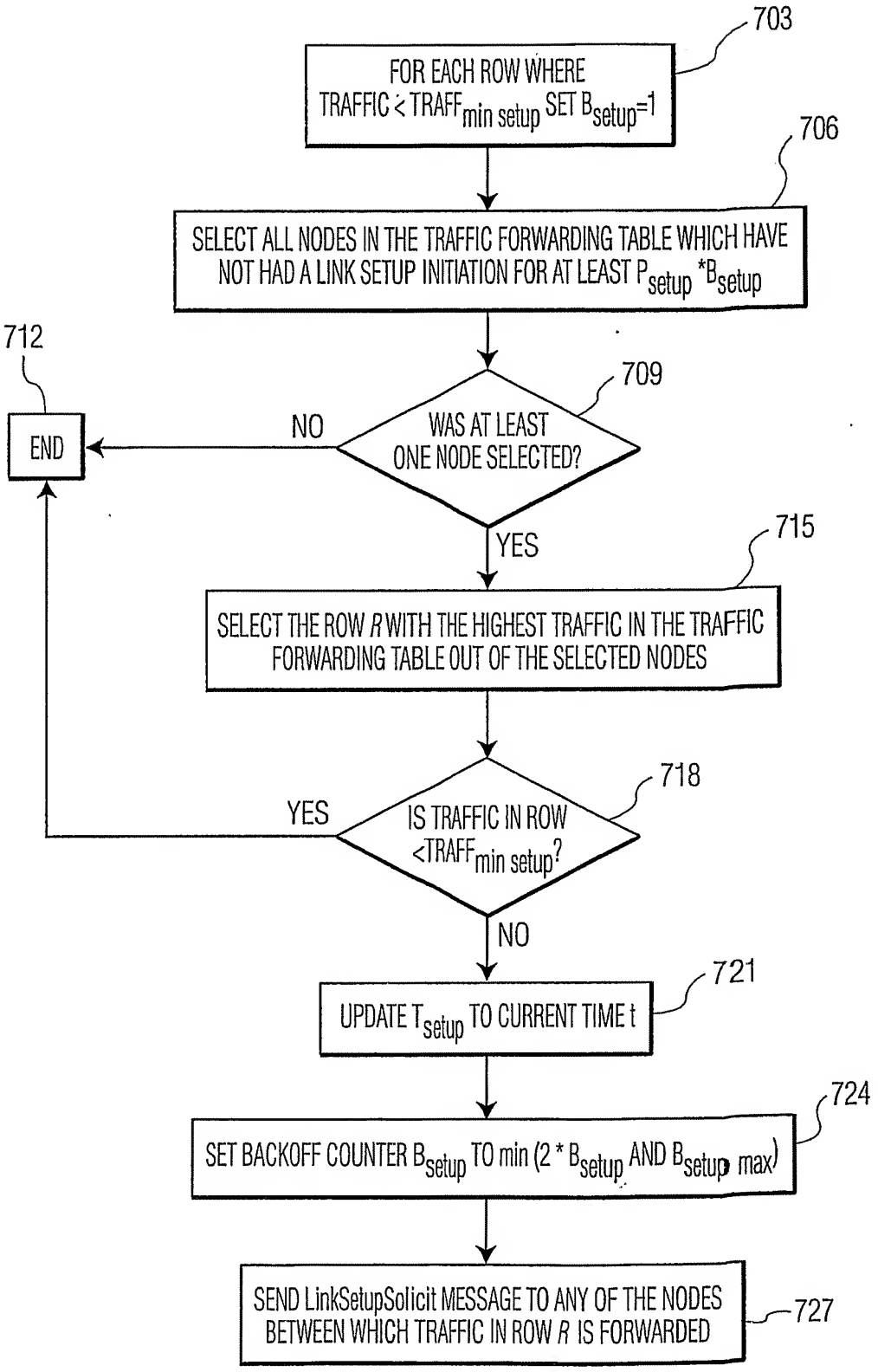

703
FOR EACH ROW WHERE TRAFFIC < TRAFF min setup SET B setup=1
706
SELECT ALL NODES IN THE TRAFFIC FORWARDING TABLE WHICH HAVE NOT HAD A LINK SETUP INITIATION FOR AT LEAST P setup *B setup
712
END
709
NO
WAS AT LEAST ONE NODE SELECTED?
YES
715
SELECT THE ROW R WITH THE HIGHEST TRAFFIC IN THE TRAFFIC FORWARDING TABLE OUT OF THE SELECTED NODES
718
YES
IS TRAFFIC IN ROW <TRAFF min setup?
NO
721
UPDATE T setup TO CURRENT TIME t
724
SET BACKOFF COUNTER B setup TO min (2 * B setup AND B setup, max)
SEND LinkSetupSolicit MESSAGE TO ANY OF THE NODES BETWEEN WHICH TRAFFIC IN ROW R IS FORWARDED
727

FIG. 7A

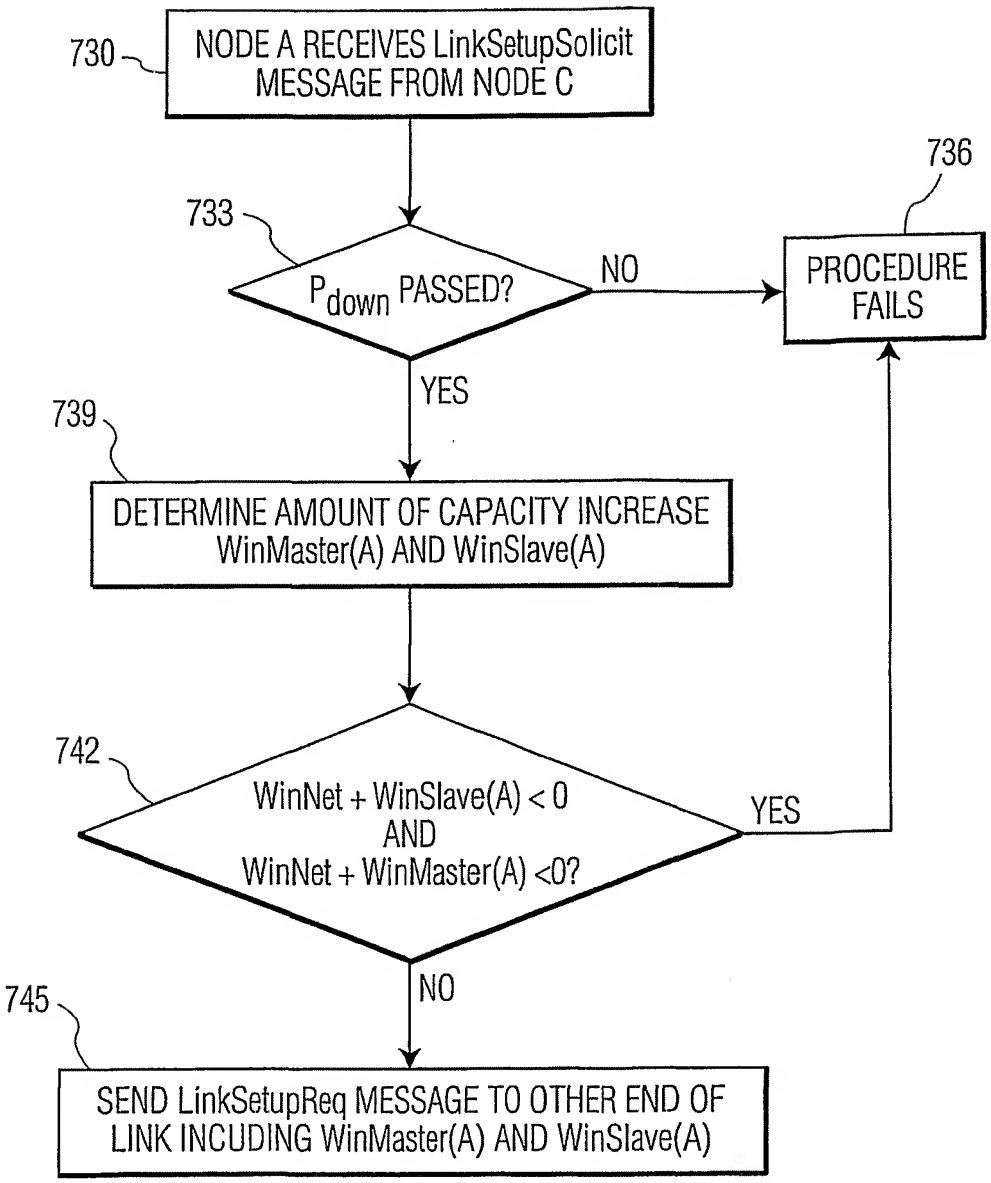

730
NODE A RECEIVES LinkSetupSolicit MESSAGE FROM NODE C
733
P_{down} PASSED?
NO
736
PROCEDURE FAILS
YES
739
DETERMINE AMOUNT OF CAPACITY INCREASE WinMaster(A) AND WinSlave(A)
742
WinNet + WinSlave(A) < 0 AND WinNet + WinMaster(A) <0?
YES
NO
745
SEND LinkSetupReq MESSAGE TO OTHER END OF LINK INCUDING WinMaster(A) AND WinSlave(A)

FIG. 7B

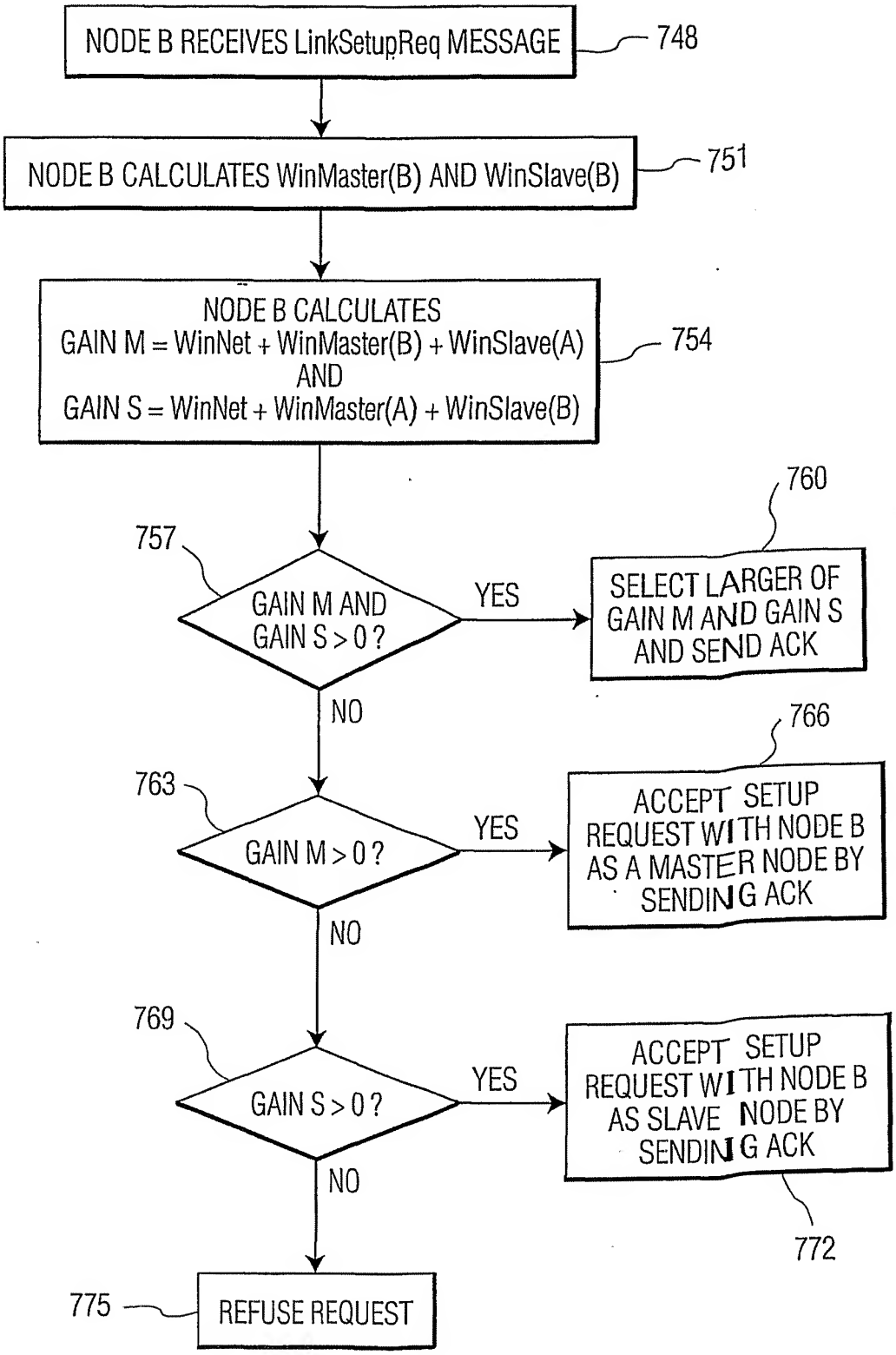

NODE B RECEIVES LinkSetupReq MESSAGE
748
NODE B CALCULATES WinMaster(B) AND WinSlave(B)
751
NODE B CALCULATES
GAIN M = WinNet + WinMaster(B) + WinSlave(A)
AND
GAIN S = WinNet + WinMaster(A) + WinSlave(B)
754
757
GAIN M AND GAIN S > 0 ?
YES
760
SELECT LARGER OF GAIN M AND GAIN S AND SEND ACK
NO
763
GAIN M > 0 ?
YES
766
ACCEPT SETUP REQUEST WITH NODE B AS A MASTER NODE BY SENDING ACK
NO
769
GAIN S > 0 ?
YES
ACCEPT SETUP REQUEST WITH NODE B AS SLAVE NODE BY SENDING ACK
772
NO
775
REFUSE REQUEST

FIG. 7C

LinkSetupSolicit	FROM	TO
BETWEEN	WinNet	

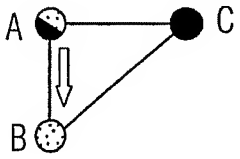
FIG. 8A

LinkSetupReq		FROM		TO
TRAFF	M/S	WinNet	WinMaster	WinSlave

FIG. 8B

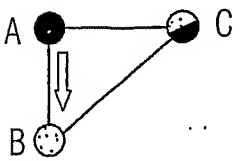
LinkSetupAck	FROM	TO
M/S	INFO	

FIG. 8C



TRAFFIC:
A→B: 100

FIG. 9A



TRAFFIC:
A→B 100

FIG. 9B

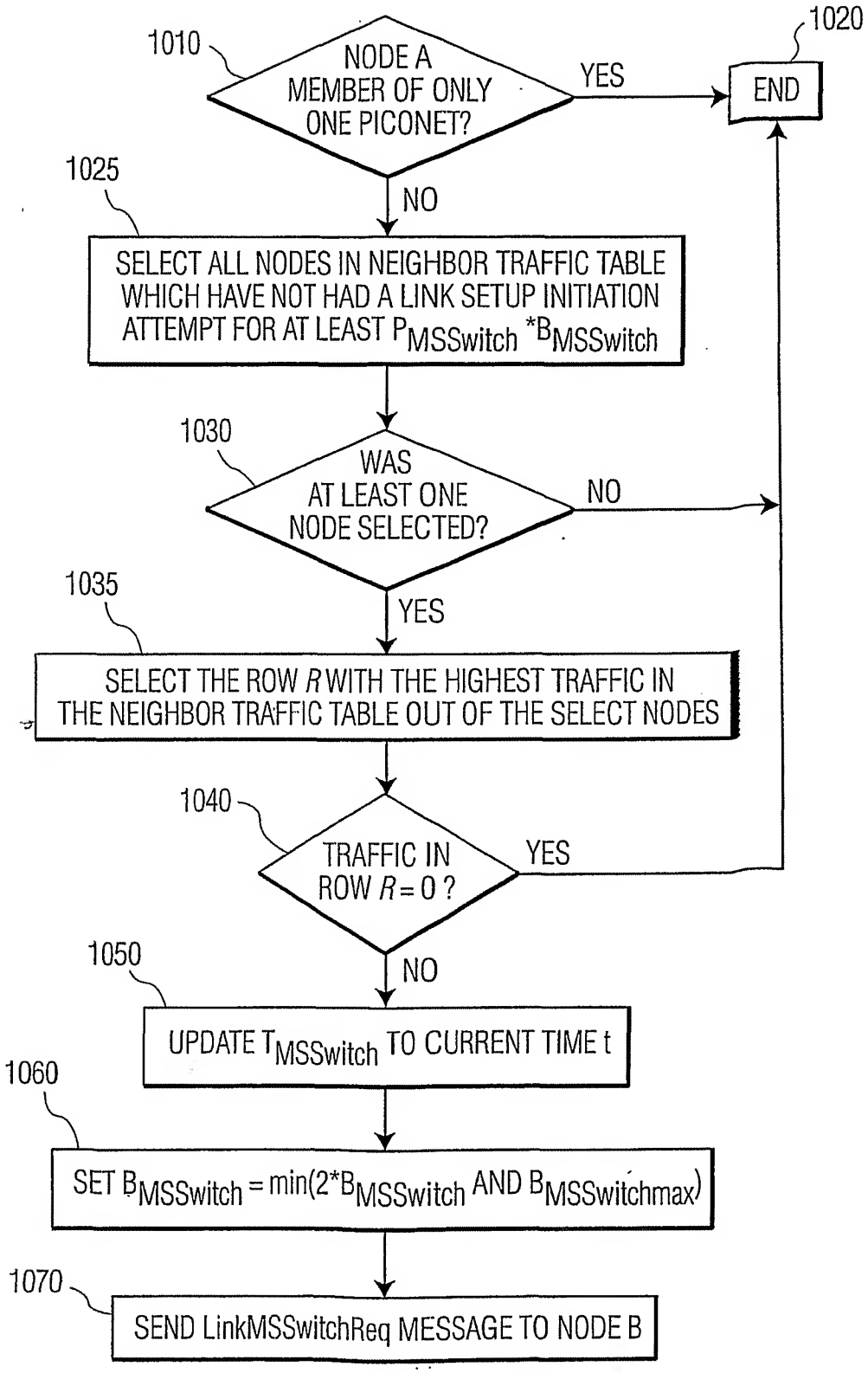

1010
NODE A MEMBER OF ONLY ONE PICONET?
YES
1020
END
NO
1025
SELECT ALL NODES IN NEIGHBOR TRAFFIC TABLE WHICH HAVE NOT HAD A LINK SETUP INITIATION ATTEMPT FOR AT LEAST $P_{MSSwitch}$ *$B_{MSSwitch}$
1030
WAS AT LEAST ONE NODE SELECTED?
NO
YES
1035
SELECT THE ROW *R* WITH THE HIGHEST TRAFFIC IN THE NEIGHBOR TRAFFIC TABLE OUT OF THE SELECT NODES
1040
TRAFFIC IN ROW *R* = 0 ?
YES
NO
1050
UPDATE $T_{MSSwitch}$ TO CURRENT TIME t
1060
SET $B_{MSSwitch}$ = min(2*$B_{MSSwitch}$ AND $B_{MSSwitchmax}$)
1070
SEND LinkMSSwitchReq MESSAGE TO NODE B

FIG. 10

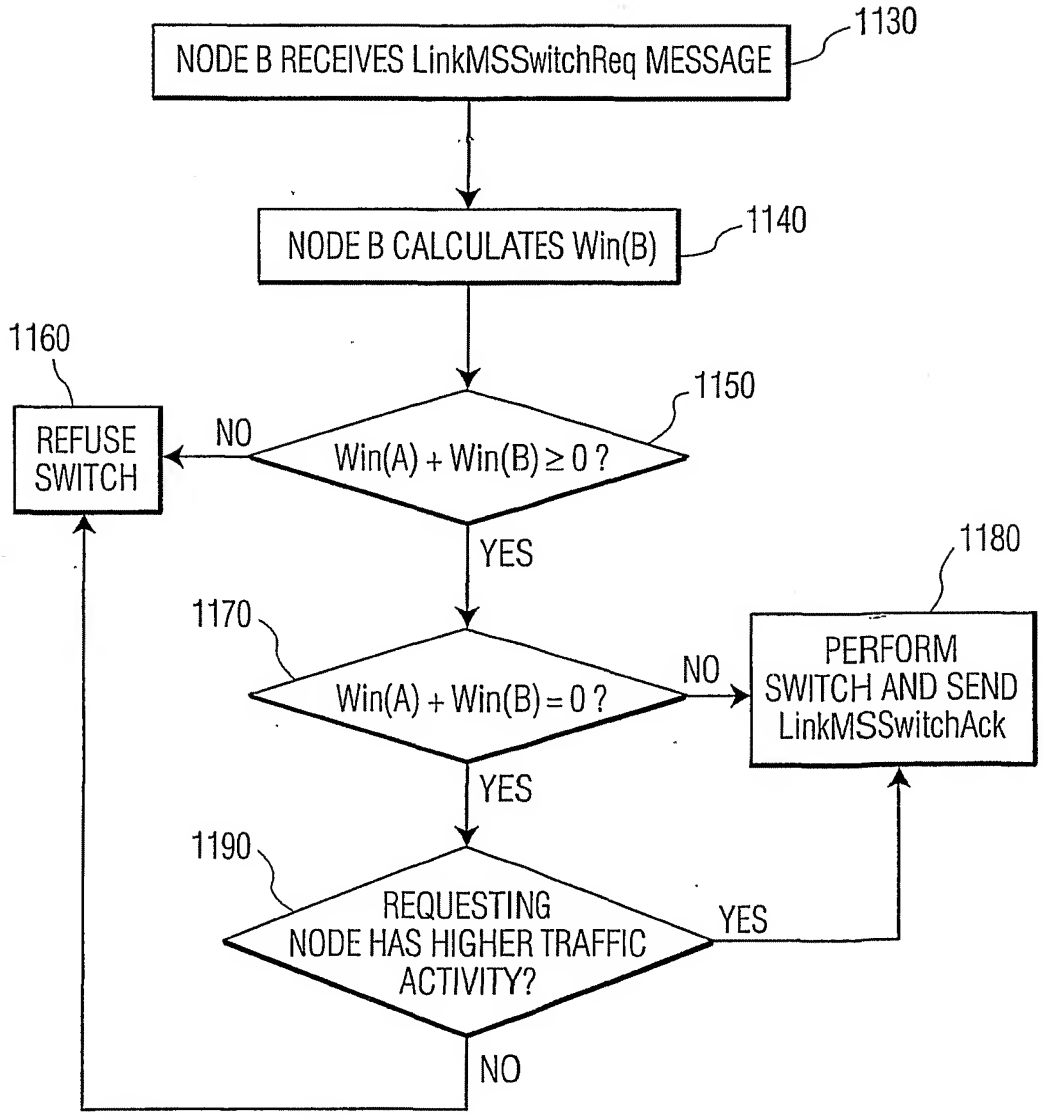

NODE B RECEIVES LinkMSSwitchReq MESSAGE
1130
NODE B CALCULATES Win(B)
1140
1160
1150
REFUSE SWITCH
NO
Win(A) + Win(B) ≥ 0 ?
YES
1180
1170
Win(A) + Win(B) = 0 ?
NO
PERFORM SWITCH AND SEND LinkMSSwitchAck
YES
1190
REQUESTING NODE HAS HIGHER TRAFFIC ACTIVITY?
YES
NO

FIG. 11

LinkMSSwitchReq		FROM	TO
TRAFF	M/S	Win	

FIG. 12A

LinkMSSwitchAck	FROM	TO
INFO		

FIG. 12B

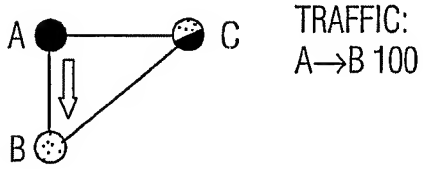


FIG. 13A

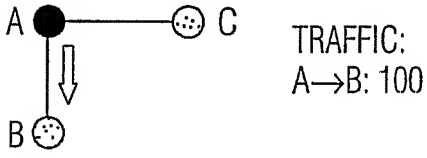


FIG. 13B

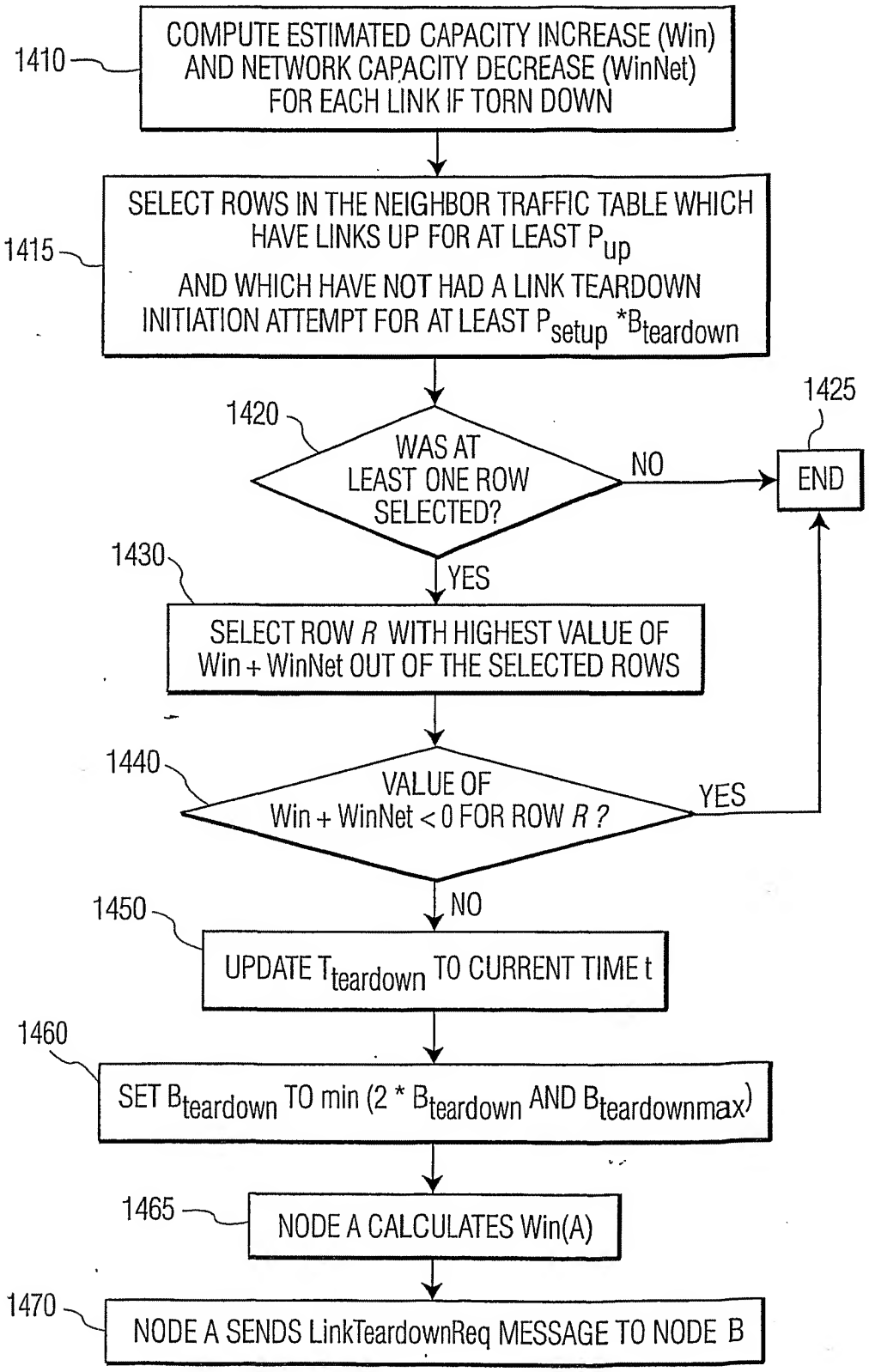

1410
COMPUTE ESTIMATED CAPACITY INCREASE (Win) AND NETWORK CAPACITY DECREASE (WinNet) FOR EACH LINK IF TORN DOWN
1415
SELECT ROWS IN THE NEIGHBOR TRAFFIC TABLE WHICH HAVE LINKS UP FOR AT LEAST P_{up} AND WHICH HAVE NOT HAD A LINK TEARDOWN INITIATION ATTEMPT FOR AT LEAST P_{setup} *$B_{teardown}$
1420
WAS AT LEAST ONE ROW SELECTED?
NO
1425
END
YES
1430
SELECT ROW *R* WITH HIGHEST VALUE OF Win + WinNet OUT OF THE SELECTED ROWS
1440
VALUE OF Win + WinNet < 0 FOR ROW *R* ?
YES
NO
1450
UPDATE $T_{teardown}$ TO CURRENT TIME t
1460
SET $B_{teardown}$ TO min (2 * $B_{teardown}$ AND $B_{teardownmax}$)
1465
NODE A CALCULATES Win(A)
1470
NODE A SENDS LinkTeardownReq MESSAGE TO NODE B

FIG. 14

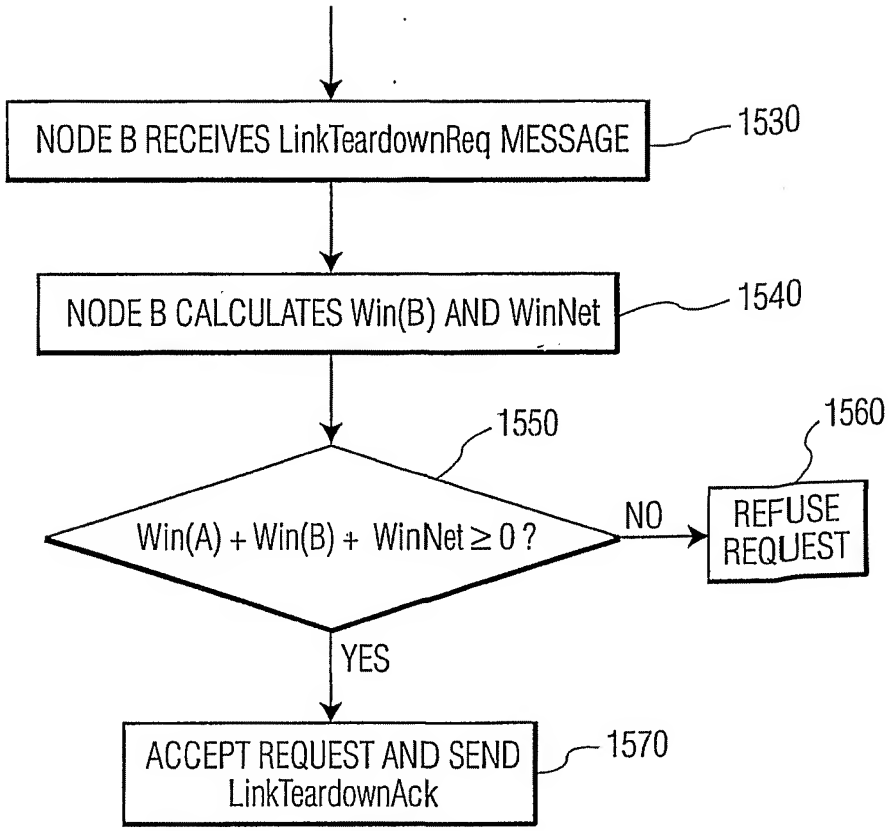


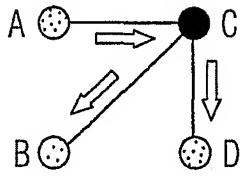
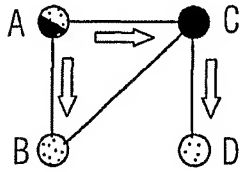
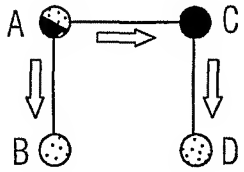
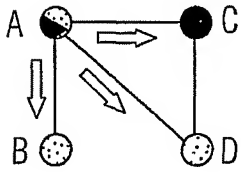
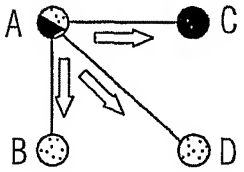
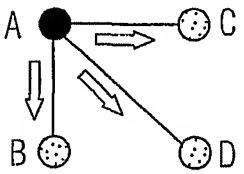
FIG. 15

LinkTeardownReq		FROM	TO
TRAFF	Win	WinNet	

FIG. 16A

LinkTeardownAck	FROM	TO	INFO

FIG. 16B

NETWORK	TRAFFIC	ACTION
	A→C: 100 A→C→B: 100 A→C→D: 100	C: LinkSetupSolicit A-B A: LinkSetupReq A-B B: LinkSetupAck A-B *Set up A-B*
FIG. 17A		
	A→C: 100 A→B: 100 A→C→D: 100	B: LinkTeardownReq B-C C: LinkTeardownAck B-C *Teardown B-C*
FIG. 17B		
	A→C: 100 A→B: 100 A→C→D: 100	C: LinkSetupSolicit A-D A: LinkSetupReq A-D D: LinkSetupAck A-D *Set up A-D*
FIG. 17C		
	A→C: 100 A→B: 100 A→D: 100	D: LinkTeardownReq C-D C: LinkTeardownAck C-D *Teardown C-D*
FIG. 17D		
	A→C: 100 A→B: 100 A→D: 100	A: LinkMSSwitchReq A-C C: LinkMSSwitchAck A-C *LinkMSSwitch A-C*
FIG. 17E		
	A→C: 100 A→B: 100 A→D: 100	
FIG. 17F		

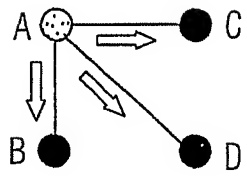
NETWORK	TRAFFIC	ACTION
	A→C: 100 A→B: 100 A→D: 100	A: LinkMSSwitchReq A-C C: LinkMSSwitchAck A-C *LinkMSSwitch A-C*

FIG. 18A

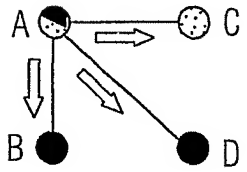
	A→C: 100 A→B: 100 A→D: 100	A: LinkMSSwitchReq A-D D: LinkMSSwitchAck A-D *LinkMSSwitch A-D*

FIG. 18B

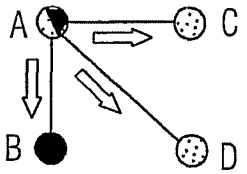
	A→C: 100 A→B: 100 A→D: 100	A: LinkMSSwitchReq A-B B: LinkMSSwitchAck A-B *LinkMSSwitch A-B*

FIG. 18C

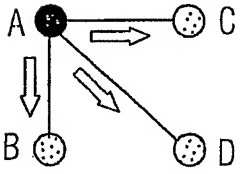
	A→C: 100 A→B: 100 A→D: 100	

FIG. 18D

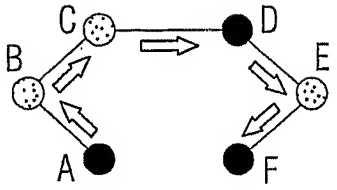
NETWORK	TRAFFIC	ACTION
	A→B: 100 A→B→C: 100 A→B→C→D: 100 A→B→C→D→E: 100 A→B→C→D→E→F: 100	B: LinkSetup A-C C: LinkSetup B-D E: LinkSetup D-F

FIG. 19A

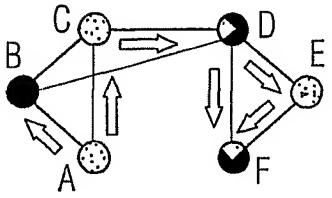
	A→B: 100 A→C: 100 A→C→D: 100 A→C→D→E: 100 A→C→D→F: 100	C: LinkTeardown B-C D: LinkTeardown B-D E: LinkTeardown E-F

FIG. 19B

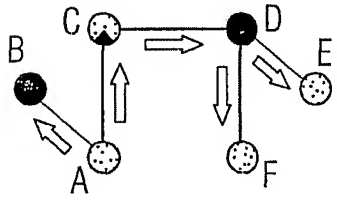
	A→B: 100 A→C: 100 A→C→D: 100 A→C→D→E: 100 A→C→D→F: 100	C: LinkMSswitch C-D A: LinkMSswitch A-B

FIG. 19C

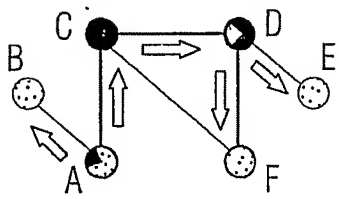
	A→B: 100 A→C: 100 A→C→D: 100 A→C→D→E: 100 A→C→D→F: 100	C: LinkSetup A-D D: LinkSetup C-F

FIG. 19D

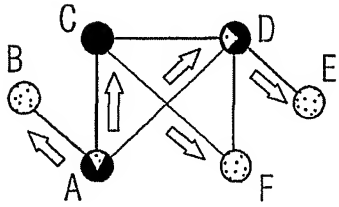
	A→B: 100 A→C: 100 A→D: 100 A→D→E: 100 A→C→F: 100	D: LinkTeardown C-D F: LinkTeardown D-F

FIG. 19E

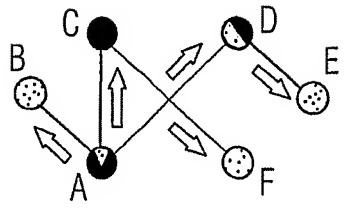
NETWORK	TRAFFIC	ACTION
	A→B: 100 A→C: 100 A→D: 100 A→D→E: 100 A→C→F: 100	A: LinkMSswitch A-C

FIG. 19F

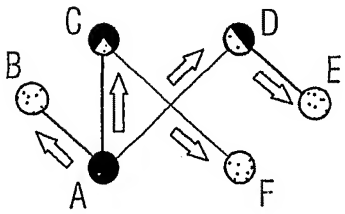
	A→B: 100 A→C: 100 A→D: 100 A→D→E: 100 A→C→F: 100	C: LinkSetup A-F D: LinkSetup A-E

FIG. 19G

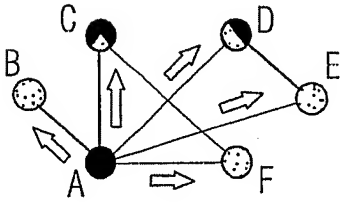
	A→B: 100 A→C: 100 A→D: 100 A→E: 100 A→F: 100	F: LinkTeardown C-F D: LinkTeardown D-E

FIG. 19H

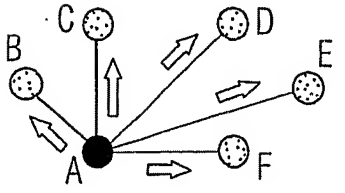
	A→B: 100 A→C: 100 A→D: 100 A→E: 100 A→F: 100

FIG. 19I

INTERNATIONAL SEARCH REPORT

International Application No
PCT/SE 01/01681

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/56

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 026 297 A (HAARTSEN JACOBUS CORNELIS) 15 February 2000 (2000-02-15) column 3, line 66 -column 7, line 27	1,15
A	WO 99 14897 A (TELEFONAKTIEBOLAGET LM ERICSSON) 25 March 1999 (1999-03-25) page 23, line 6 -page 26, line 17	1,15
A	HAARTSEN J: "DIE BLUETOOTH-ÜBERTRAGUNG" FUNKSCHAU,FRANZIS-VERLAG K.G. MÜNCHEN,DE, vol. 72, no. 15, 9 July 1999 (1999-07-09), pages 76-80, XP000913229 ISSN: 0016-2841	
A	WO 95 24081 A (PROXIM, INC.) 8 September 1995 (1995-09-08)	

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 August 2001	04/09/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Behringer, L.V.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6026297 A	15-02-2000	AU 9195098 A	05-04-1999
		BR 9812219 A	18-07-2000
		CN 1270726 T	18-10-2000
		EE 200000135 A	15-02-2001
		EP 1016242 A	05-07-2000
		PL 339345 A	18-12-2000
		WO 9914898 A	25-03-1999
WO 9914897 A	25-03-1999	AU 9099498 A	05-04-1999
		BR 9812226 A	18-07-2000
		CN 1278974 T	03-01-2001
		EP 1016241 A	05-07-2000
		NO 20001378 A	12-05-2000
WO 9524081 A	08-09-1995	AU 1974795 A	18-09-1995
		EP 0748540 A	18-12-1996